بسم الله الرحمن الرحيم

لغة النمذجة الموحدة



إعداد الطالب/ محمد أحمد سالم الوصابي

قسم /تكنولوجيا المعلومات (IT)

سنه رابعه

هندسة برمجيات ٢

د.فاضل صلاح

لغة النمذجة الموحدة UML

تـم إطــلاق Unified Modeling Language (UML) عــام ١٩٩٧ كطريقــة لوضــع مخططــات تصميم للبرمجيات بهدف:

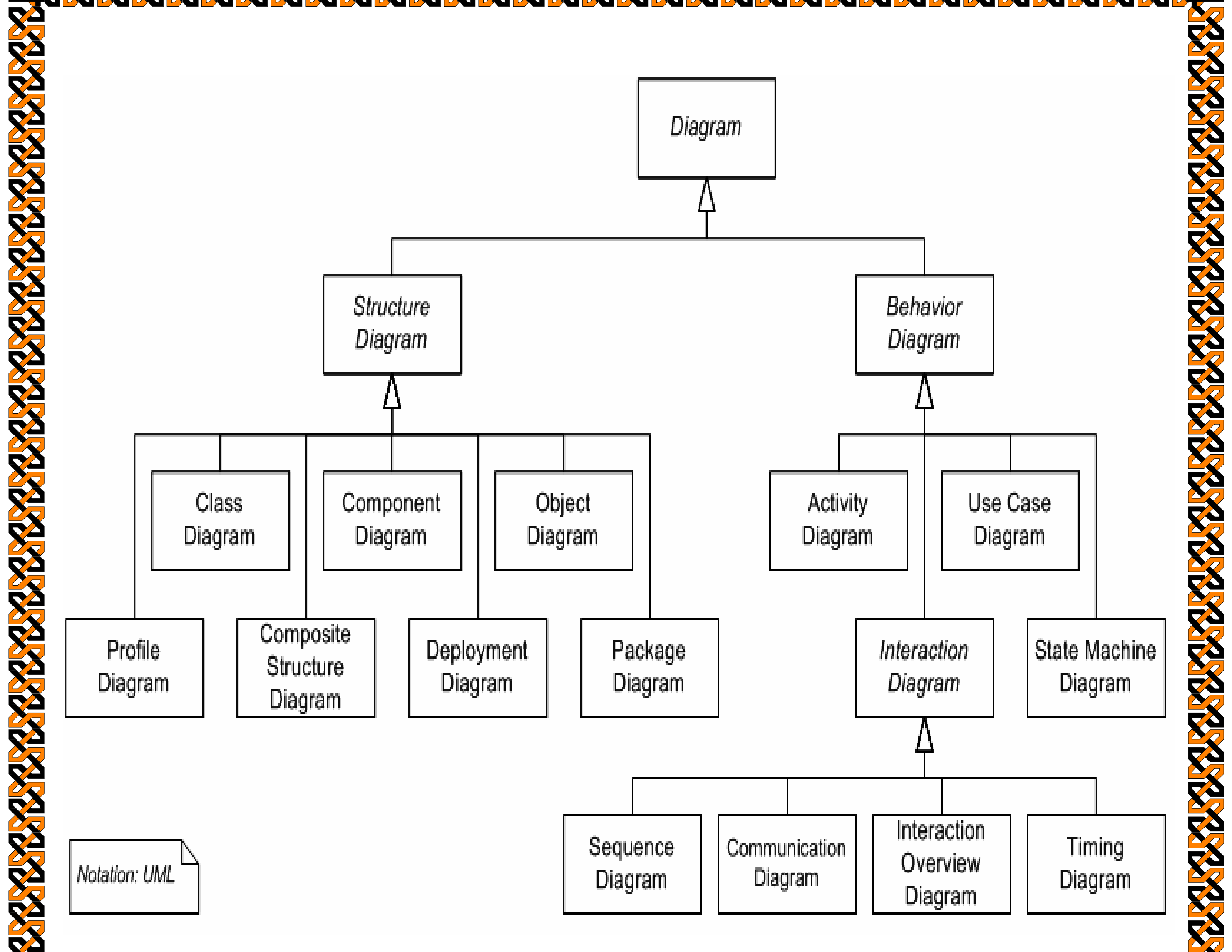
- تصميم البرمجيات بشكل احترافي.
- توثيق التصميم قبل البدء بالبرمجة.
- إعادة الاستخدام Reusability (تخفيض الكلفة).
- البرنامج الذي تم تطويره يؤدي الوظائف المطلوبة (زيادة الوثوقية).
- سهولة التعديل والصيانة وبكلفة منخفضة.
- مخططات UML تساعد المطورين على فهم النظام بسهولة وسرعة.
- لغة تواصل بين المطورين والمصممين.

تتكون UML من عناصر رسـومية توضــع ضــمن مخططــات مختلفــة لتوصــيف النظــام ، هــذه المخططــات تقــوم بتوصــيف النظــام ولــيس لهــا علاقــة بكيفيــة برمجــة هــذه الوظــائف Implementation.

تتألف الــــ UML مــن ١٤ نــوع مــن المخططــات وتقــسم إلــى نــوعين Structure Diagram, Behavior Diagram.

Structure Diagram: تركز على عناصر النظــام بــشكل مــستقل عــن الــزمن أي البينــة الــساكنة Static Structure.

Behavior Diagram: تركــز علــى وظيفيــة النظــام أي تغيراتــه مــع الــزمن Dynamic Structure.



مخطط Class diagram

الصنف Class

هو تجميع لمجموعة من الأشياء المتشابهة في خواصها أو سلوكها

مثال: (الحيوانات، وسائط النقل، النباتات ، الطلاب، .....)

الكائن Object

عند إسناد قيم للـــ Class نحصل على الكائن Object أي هو Class صفاته تملك قيم.

مثال: تحديد طالب محدد اسمه أحمد.

Class Diagram

يمكن وضع الأشياء ضمن أصناف Classes ، مهمة Class diagram توضيح هذه الأصناف والعلاقات associations فيما بينها

لكل صنف Class :

- اسم class name
- مميزات attributes
- عمليات operations وأحياناً نسميها Methods

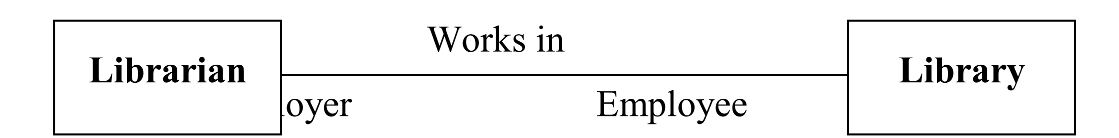
وبالتالي يمكن تمثيل Class بأحد الطرق التالية:

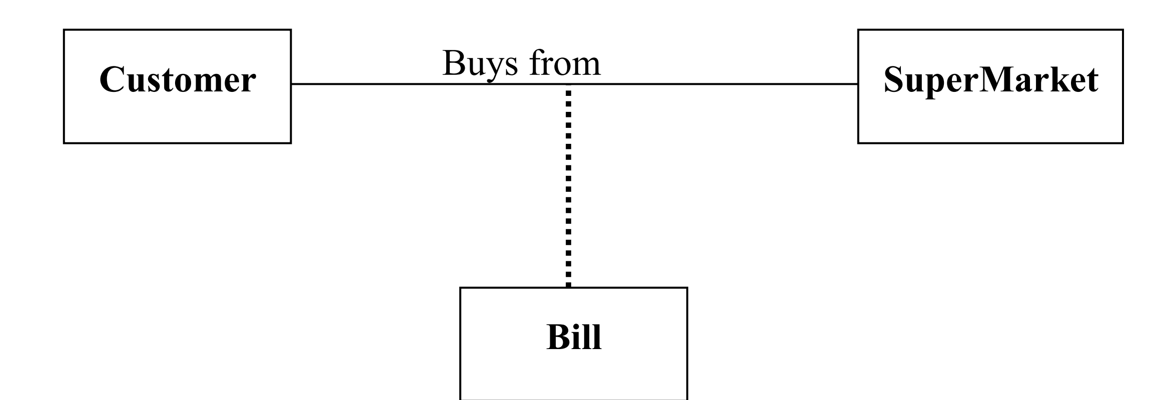
Class Name

Class Name
Att1: type 1 Att 2: type 2 = "default value" … …

Class Name
Att1: type 1 Att2: type 2 = "default value" … …
Operation1 () Operation2 (list of parameters) Operatipn3: returned value type

العلاقات بين الأصناف Associations



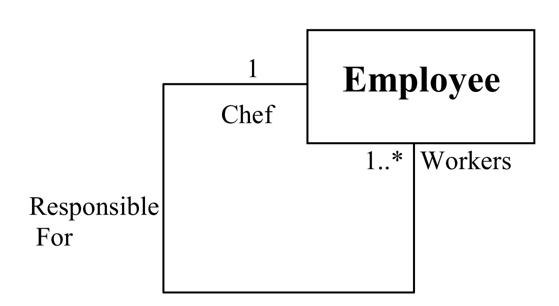


Multiplicity التعددية

يمكن ان تأخذ التعددية بين الأصناف احد الأشكال التالية:

one to one	1 ——— 1
one to many	1 ——— *
one to one or more	1 ——— 1..*
one to zero or more	1 ——— 0..*
one to exactly n	1 ——— n

في بعض الأحيان يكون هناك علاقة بين الصنف Class ونفسه وتسمى هذه العلاقة **Reflexive association**



يمكن أن تكون attributes & operations في إحدى الحالات التالية:

Public (+)

Private (–)

Protected (#)

مثال:

Inheritance & Generalization الوراثة

توجد بعض المميزات المشتركة بين الأصناف والتي يمكن نقلها بين الأصناف Classes وبالتالي يفضل في هذه الحالة أن ندخل مفهوم الوراثة.

يمكن لأحد الأصناف نسميه Child class (subclass) أن يرث attributes and operations من صنف آخر نسميه Parent class (super class) حيث يمكن أن ينوب الأب عن ابنه ولكن العكس غير صحيح.

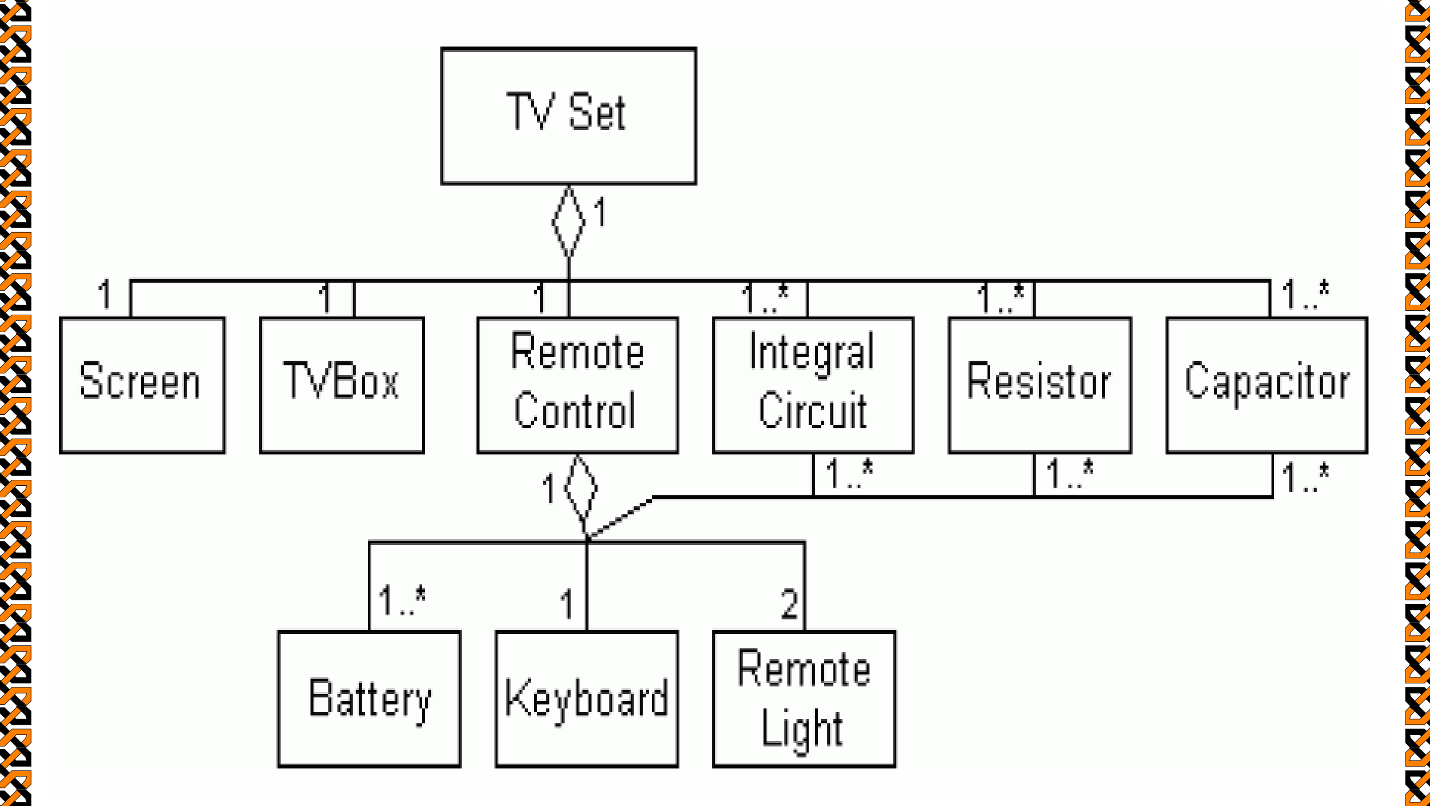
مثال: موظفين في مؤسسة ما (مدير، مهندس، سكرتيرة، رئيس قسم، عامل بوفية، عامل نظافة، ...)

Aggregations علاقة

هي علاقة بين الكل والجزء ويتم تمثيلها كما يلي:

تكون دورة حياة صف الجزء مستقلة عن دورة حياة صف الكل (يمكن رؤية الجزء بعيداً عن الكل).

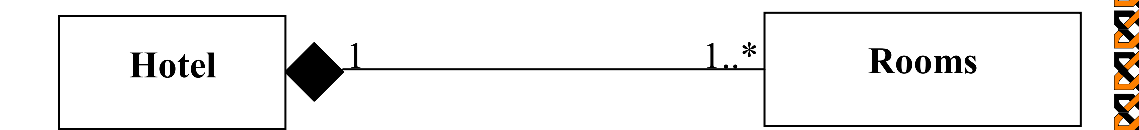
أمثلة:



علاقة *Composition*

هي علاقة بين الكل والجزء أيضاً لكنها أشد قسوة حيث أن دورة حياة صف الجزء تعتمد على دورة حياة صف الكل (لا يمكن رؤية الجزء بعيداً عن الكل).

يتم تمثيل هذه العلاقة كما يلي:

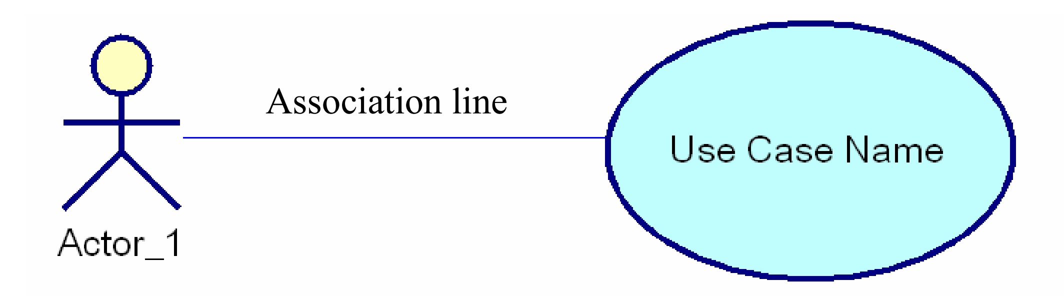


Use Case diagram

كل Use Case تمثل وظيفة من وظائف النظام وبالتالي يتضمن هذا المخطط :

- الوظائف المطلوبة من النظام.
- لمستخدم actor الذي يقوم بطلب هذه الوظيفة (قد يكون نظام آخر).

يساعدنا على بناء Use case diagram فهم سيناريو العمل والذي نحصل عليه من مستخدم النظام.

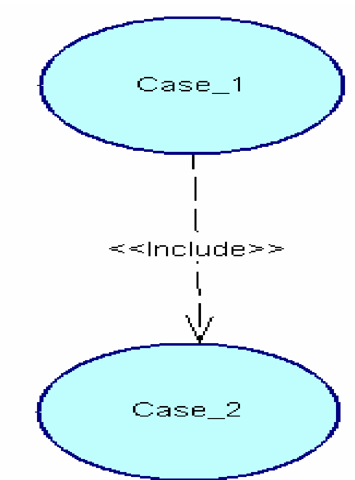


كل Use Case عبارة عن سيناريو يتكون من مجموعة من الخطوات لتنفيذ وظيفة محددة.

العلاقات في Use Case

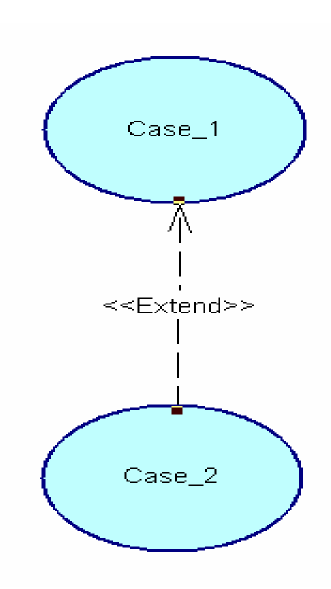
<u>Include</u>

تمكننا هذه العلاقة من إعادة استخدام الخطوات الموجودة داخل Use Case
يتم تمثيل هذه العلاقة كما يلي:

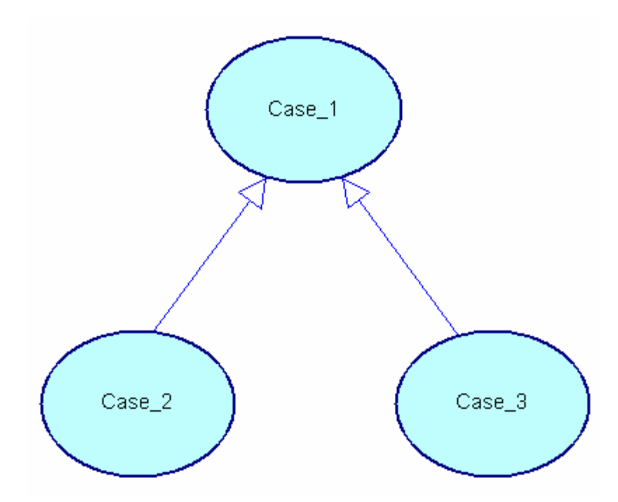


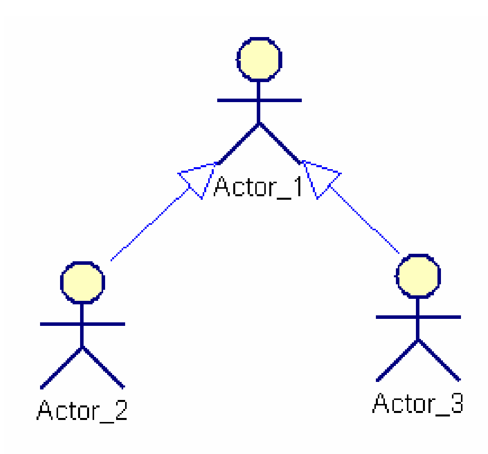
Extension

تسمح لنا هذه العلاقة بإنشاء Use Case جديدة يتم إضافتها إلى Use Case موجودة سابقا بهدف:
معالجة حالة استثنائية قد تواجهنا أو لوضع شرط على الخطوات ضمن Use Case الحالية
يتم تمثيلها كما يلي:

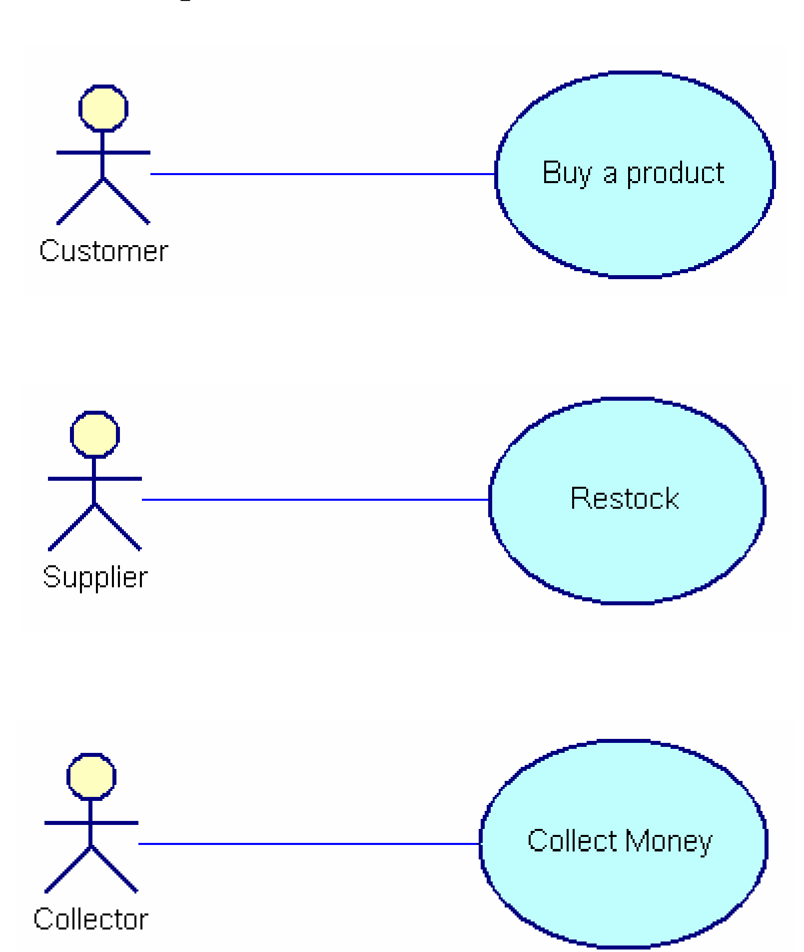


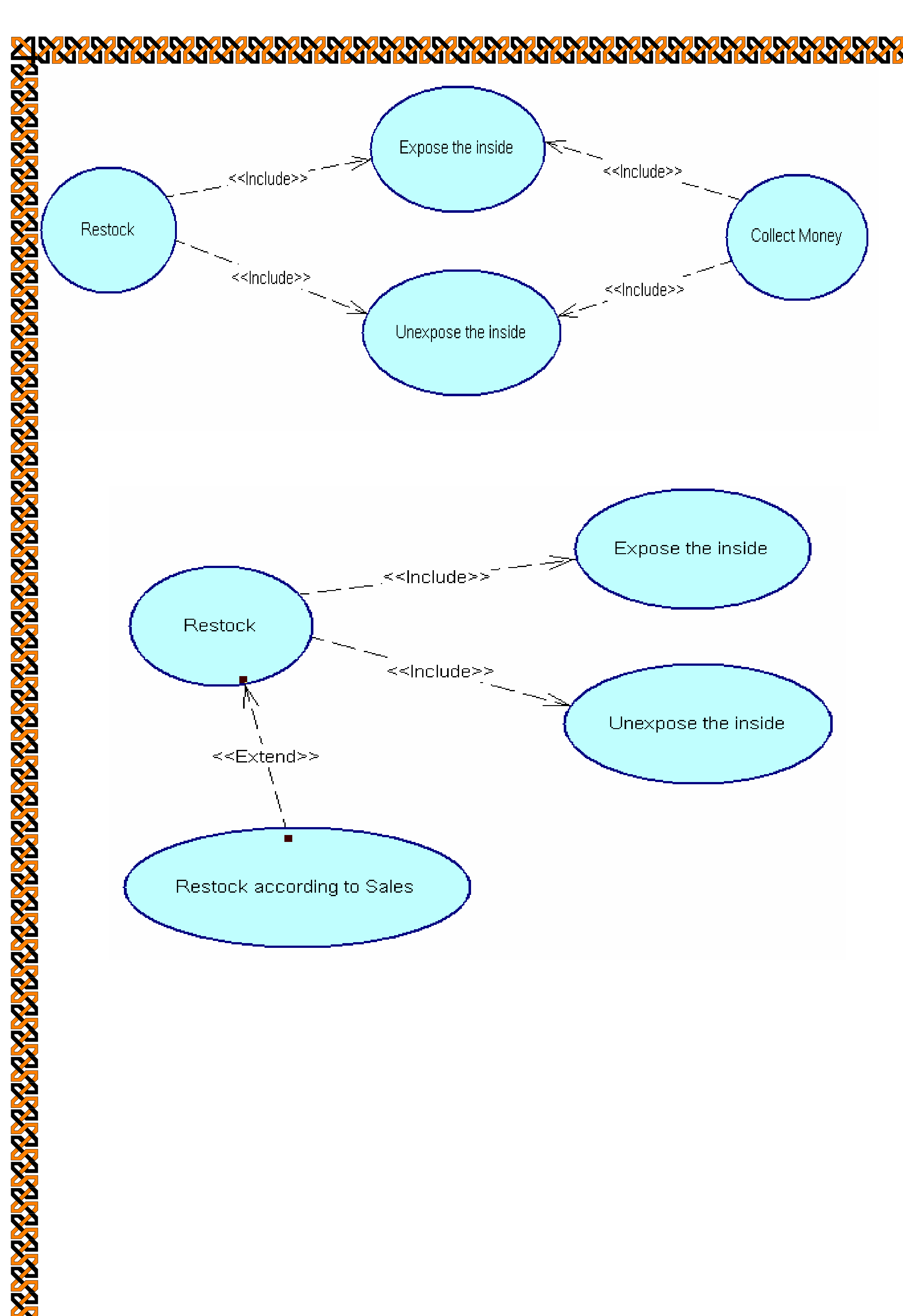
الوراثة Generalization

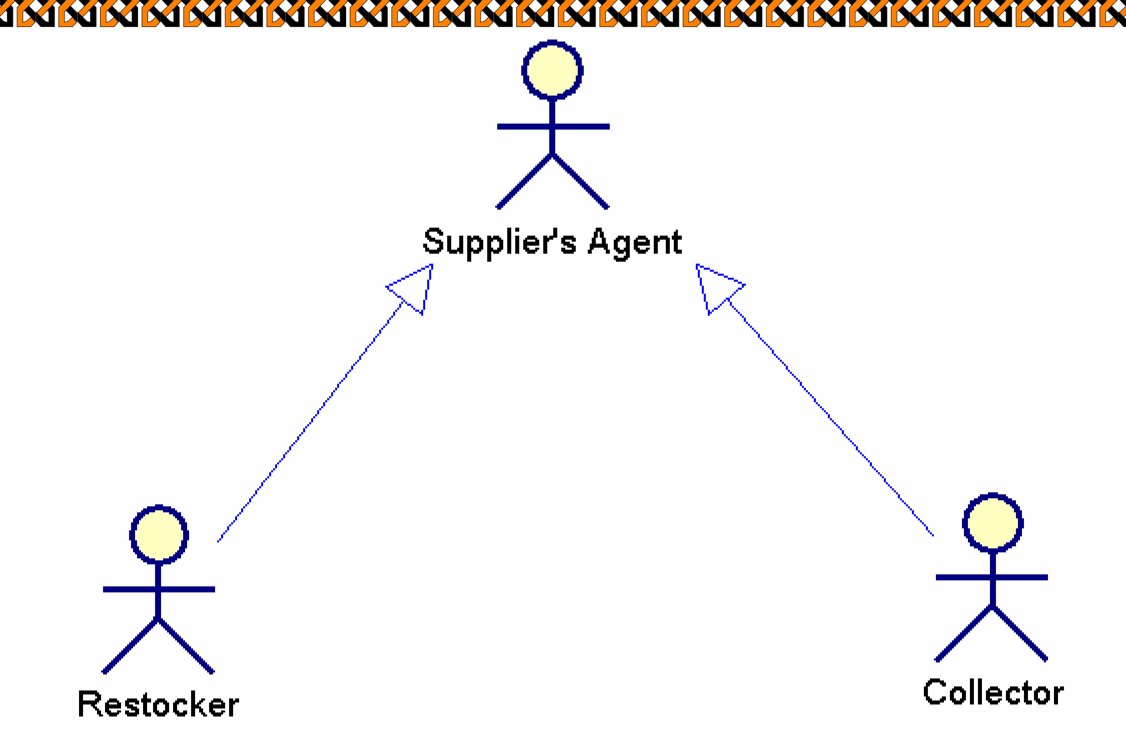




Self service Machine example



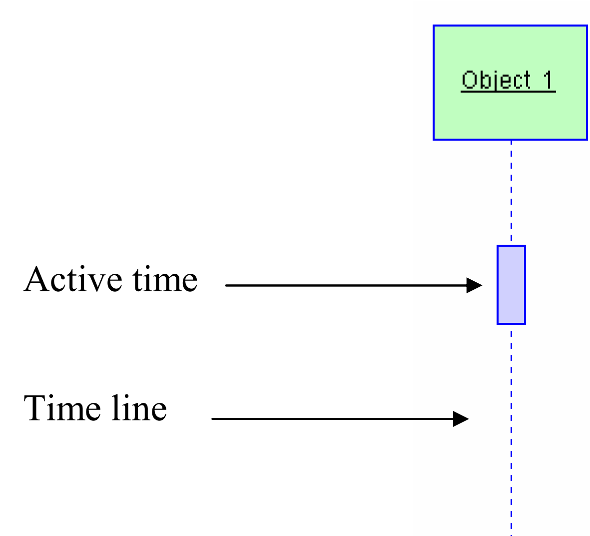

Expose the inside
<<Include>>
Restock
<<Include>>
Collect Money
<<Include>>
<<Include>>
Unexpose the inside
Expose the inside
<<Include>>
Restock
<<Include>>
Unexpose the inside
<<Extend>>
Restock according to Sales



Sequence diagram

مخطط يهدف إلى توصيف الاتصالات بين Objects عبر الزمن أي يتم إدخال بُعد الزمن Time إلى المخطط وبالتالي يتم توضيح كل التفاعلات والاتصالات بين Objects وفق تسلسل زمني.

تستخدم الرموز التالية في :Sequence diagram



توجد عدة أنواع للرسائل المتبادلة بين Objects وهي:

Message

Self message

Call message (synchronous)

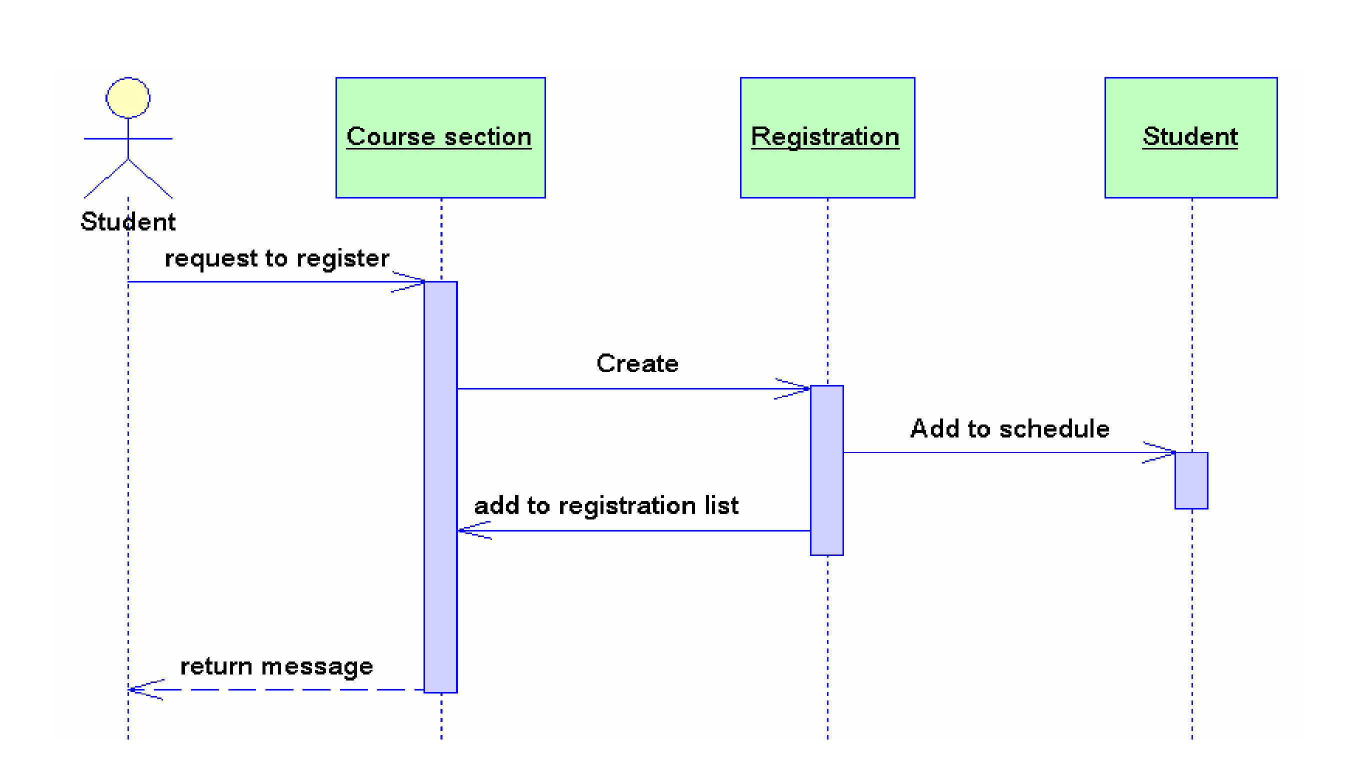
Asynchronous message

Self call message

Return message

Self return message

يوضح الشكل التالي طريقة تمثيل الرسائل في المخطط:



Actor_1

Object_1

Object_2

Message_1

Self message

<<Synchronous>>
Call message

Asynchronous message

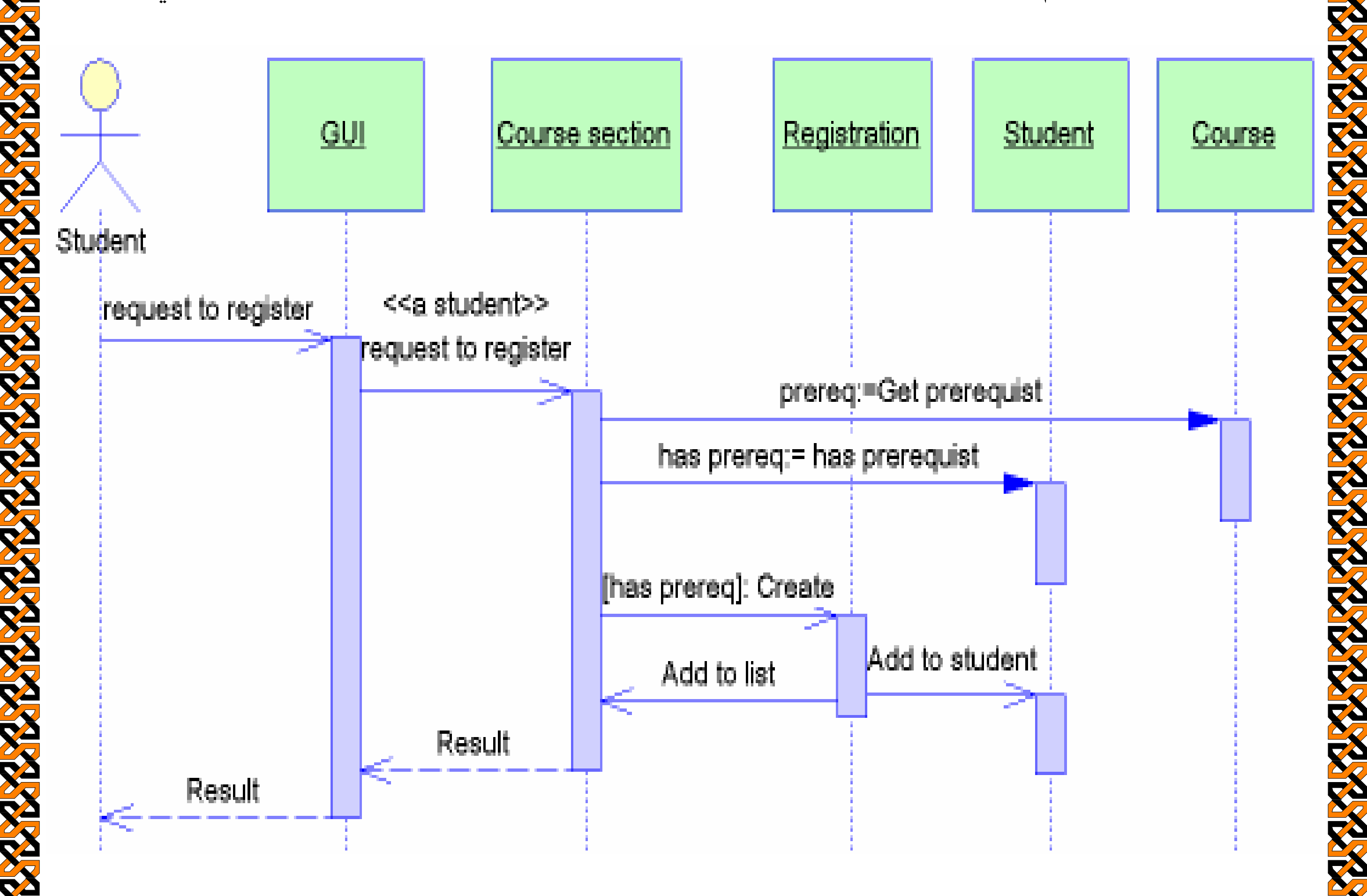
Self Call message

Return message

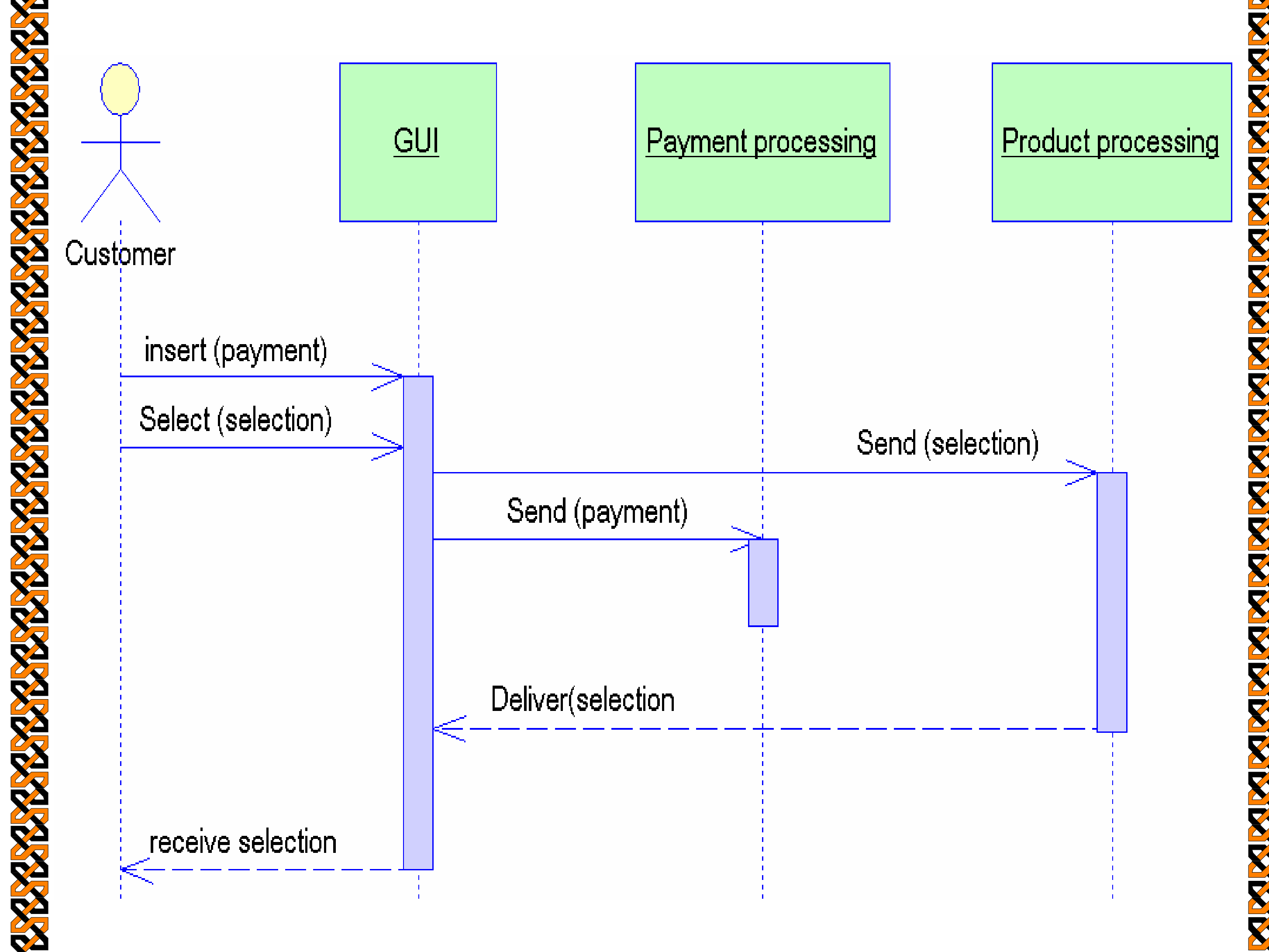
Self Return message

Return message

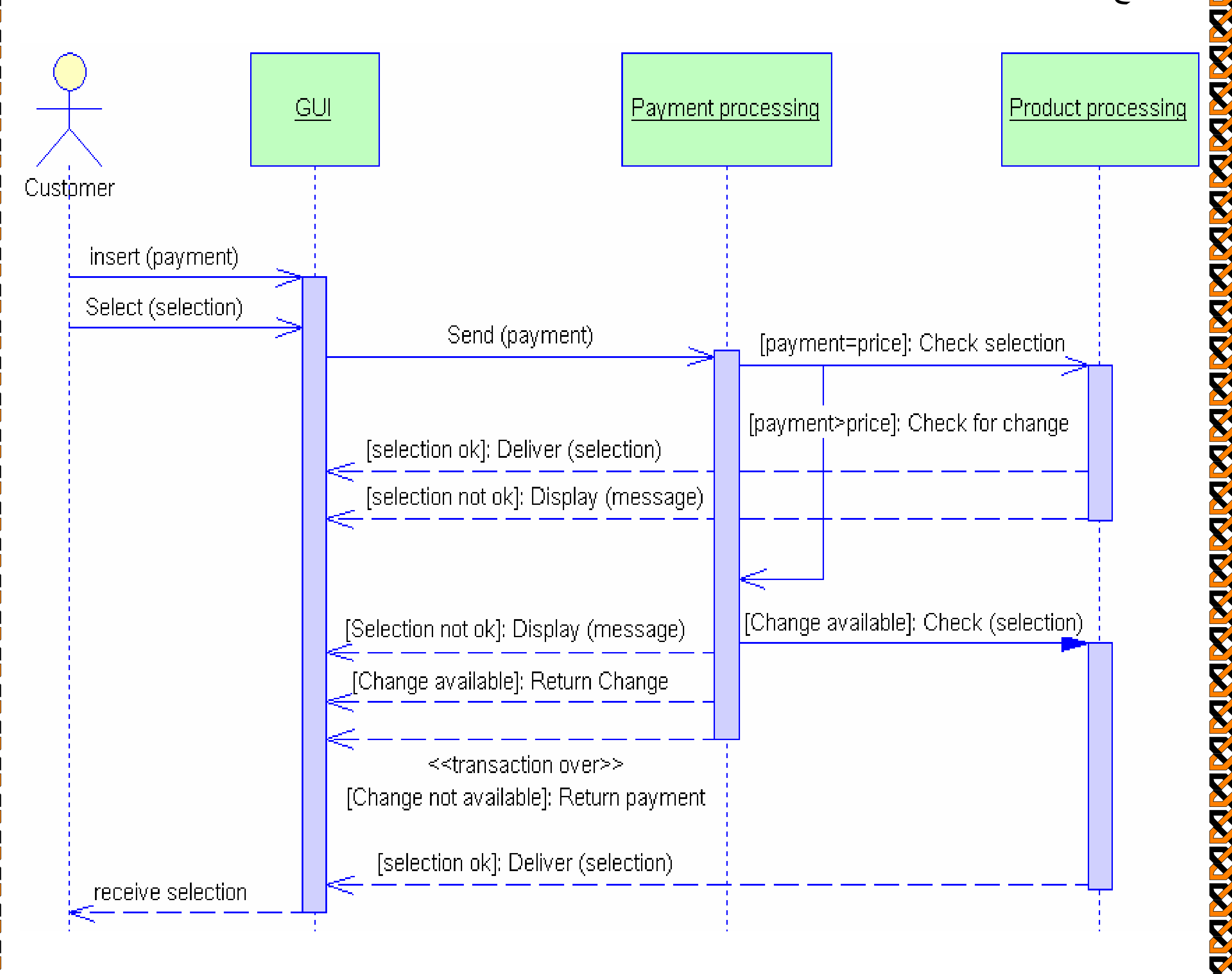
إذا أردنا تحسين النظام بحيث يأخذ بعين الاعتبار المتطلبات المسبقة لكل مادة يكون لدينا المخطط التالي:



مثال٢: في نظام self service machine يقوم الزبون بإدخال النقود ويختار منتج فتقوم الآلة بالاستجابة لطلبه وبيعه المنتج المطلوب.



يمكن تطوير النظام بجعله يأخذ بعين الاعتبار قيمة الدفعة (اكبر أو تساوي قيمة المنتج) وأيضا فيما إذا كان المنتج متوفر أو غير متوفر.



تم بحمد الله

بسم الله الرحمن الرحيم

لغة النمذجة الموحدة



إعداد الطالب/ محمد أحمد سالم الوصابي

قسم /تكنولوجيا المعلومات (١٢)

سنه رابعه

هندسة برمجيات ٢

د.فاضل صلاح

لغة النمذجة الموحدة UML

تـم إطـلاق (UML) Unified Modeling Language عـام ١٩٩٧ كطريقـة لوضـع مخططـات تصميم للبرمجيات بهدف:

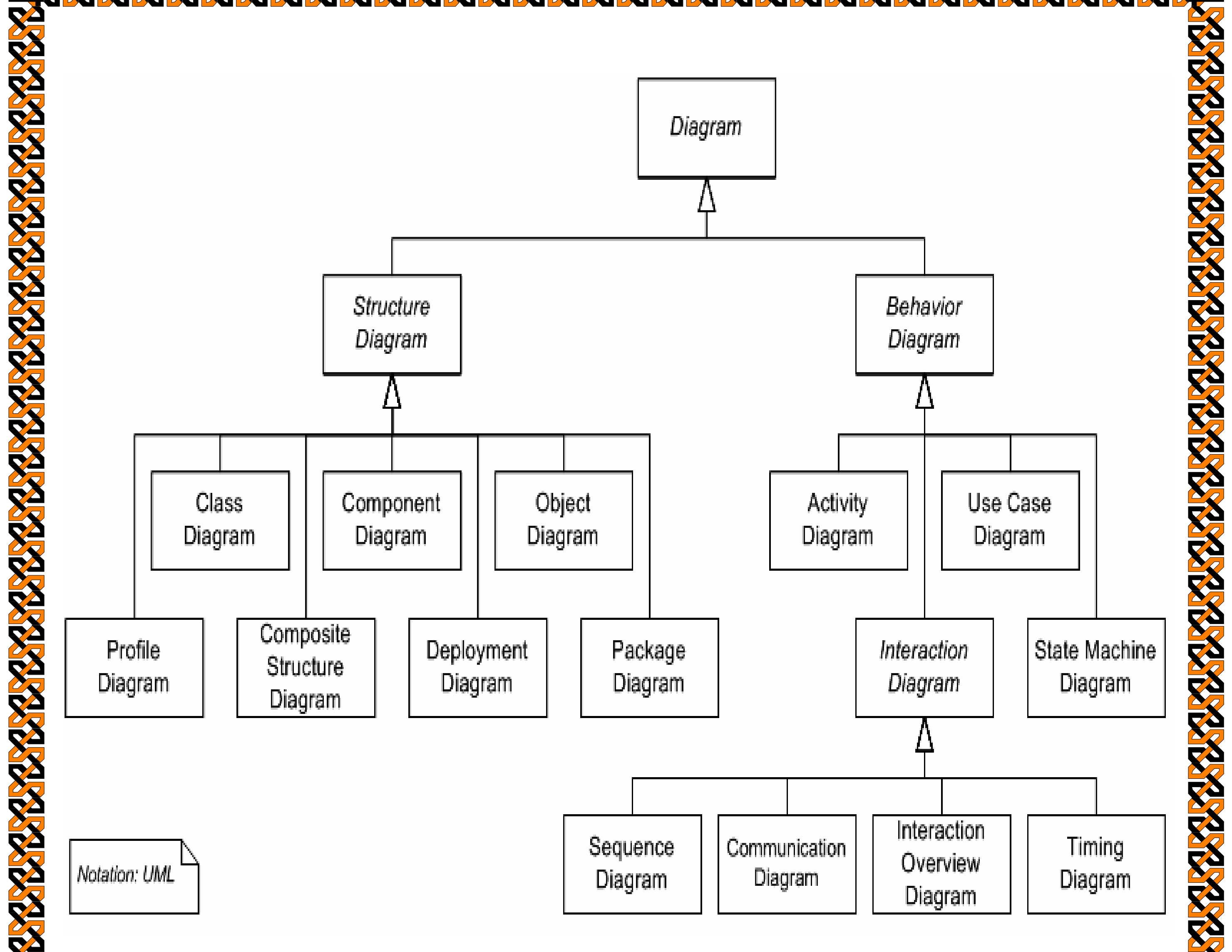
- تصميم البرمجيات بشكل احترافي.
- توثيق التصميم قبل البدء بالبرمجة.
- إعادة الاستخدام Reusability (تخفيض الكلفة).
- البرنامج الذي تم تطويره يؤدي الوظائف المطلوبة (زيادة الوثوقية).
- سهولة التعديل والصيانة وبكلفة منخفضة.
- مخططات UML تساعد المطورين على فهم النظام بسهولة وسرعة.
- لغة تواصل بين المطورين والمصممين.

تتكون UML من عناصر رسـومية توضـع ضـمن مخططـات مختلفـة لتوصـيف النظـام ، هـذه المخططـات تقـوم بتوصـيف النظـام ولـيس لهـا علاقـة بكيفيـة برمجـة هـذه الوظـائف Implementation.

تتألف الـــ UML مــن ١٤ نــوع مــن المخططــات وتقـسم إلــى نــوعين Structure Diagram, Behavior Diagram.

Structure Diagram: تركز على عناصر النظـام بـشكل مـستقل عـن الـزمن أي البيئـة الـساكنة Static Structure.

Behavior Diagram: تركـز علـى وظيفيـة النظـام أي تغيراتـه مـع الـزمن Dynamic Structure.



مخطط Class diagram

الصنف Class

هو تجميع لمجموعة من الأشياء المتشابهة في خواصها أو سلوكها

مثال: (الحيوانات، وسائط النقل، النباتات ، الطلاب، .....)

الكائن Object

عند إسناد قيم للـــ Class نحصل على الكائن Object أي هو Class صفاته تملك قيم.

مثال: تحديد طالب محدد اسمه أحمد.

Class Diagram

يمكن وضع الأشياء ضــمن أصــناف Classes ، مهمــة Class diagram توضــيح هــذه الأصــناف والعلاقات associations فيما بينها

لكل صنف Class :

- اسم class name
- مميزات attributes
- عمليات operations وأحياناً نسميها Methods

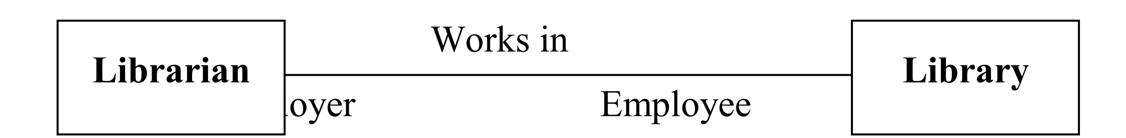
وبالتالي يمكن تمثيل Class بأحد الطرق التالية:

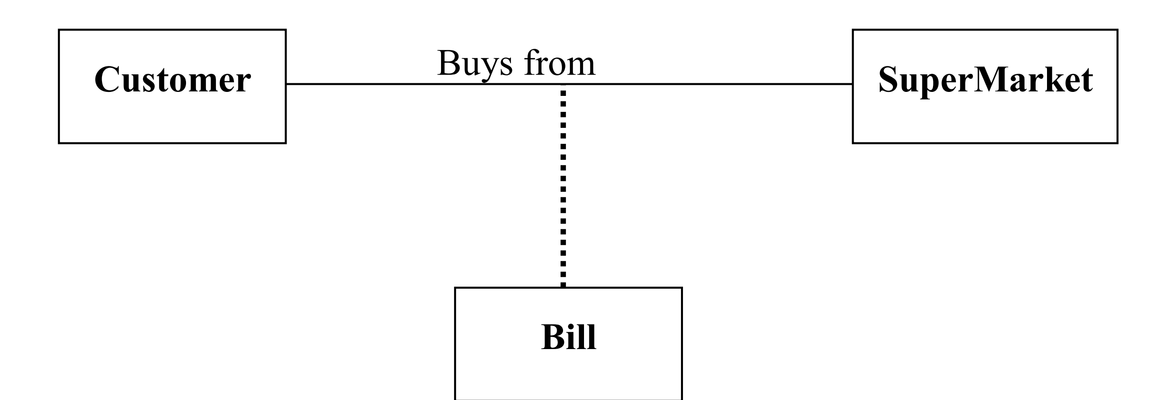
Class Name

Class Name
Att1: type 1 Att 2: type 2 = "default value" … …

Class Name
Att1: type 1 Att2: type 2 = "default value" … …
Operation1 () Operation2 (list of parameters) Operatipn3: returned value type

العلاقات بين الأصناف Associations



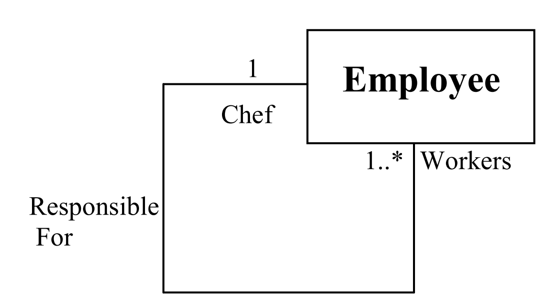


Multiplicity التعددية

يمكن ان تأخذ التعددية بين الأصناف احد الأشكال التالية:

one to one	1	1
one to many	1	*
one to one or more	1	1..*
one to zero or more	1	0..*
one to exactly n	1	n

في بعض الأحيان يكون هناك علاقة بين الصنف Class ونفسه وتسمى هذه العلاقة **Reflexive association**



يمكن أن تكون attributes & operations في إحدى الحالات التالية:

Public (+)

Private (–)

Protected (#)

مثال:

HardDisk

+modelName
+capacity
+producer
. . .

+read()
+write()
-adjustHeads()
. . .

الوراثة Inheritance & Generalization

توجد بعض المميزات المشتركة بين الأصناف والتي يمكن نقلها بين الأصناف Classes وبالتالي يفضل في هذه الحالة أن ندخل مفهوم الوراثة.

يمكن لأحد الأصناف نسميه Child class (subclass) أن يرث attributes and operations من صنف آخر نسميه Parent class (super class) حيث يمكن أن ينوب الأب عن ابنه ولكن العكس غير صحيح.

مثال: موظفين في مؤسسة ما (مدير، مهندس، سكرتيرة، رئيس قسم، عامل بوفية، عامل نظافة، ...)

Super class
Parent

Sub Class
Child

علاقة Aggregations

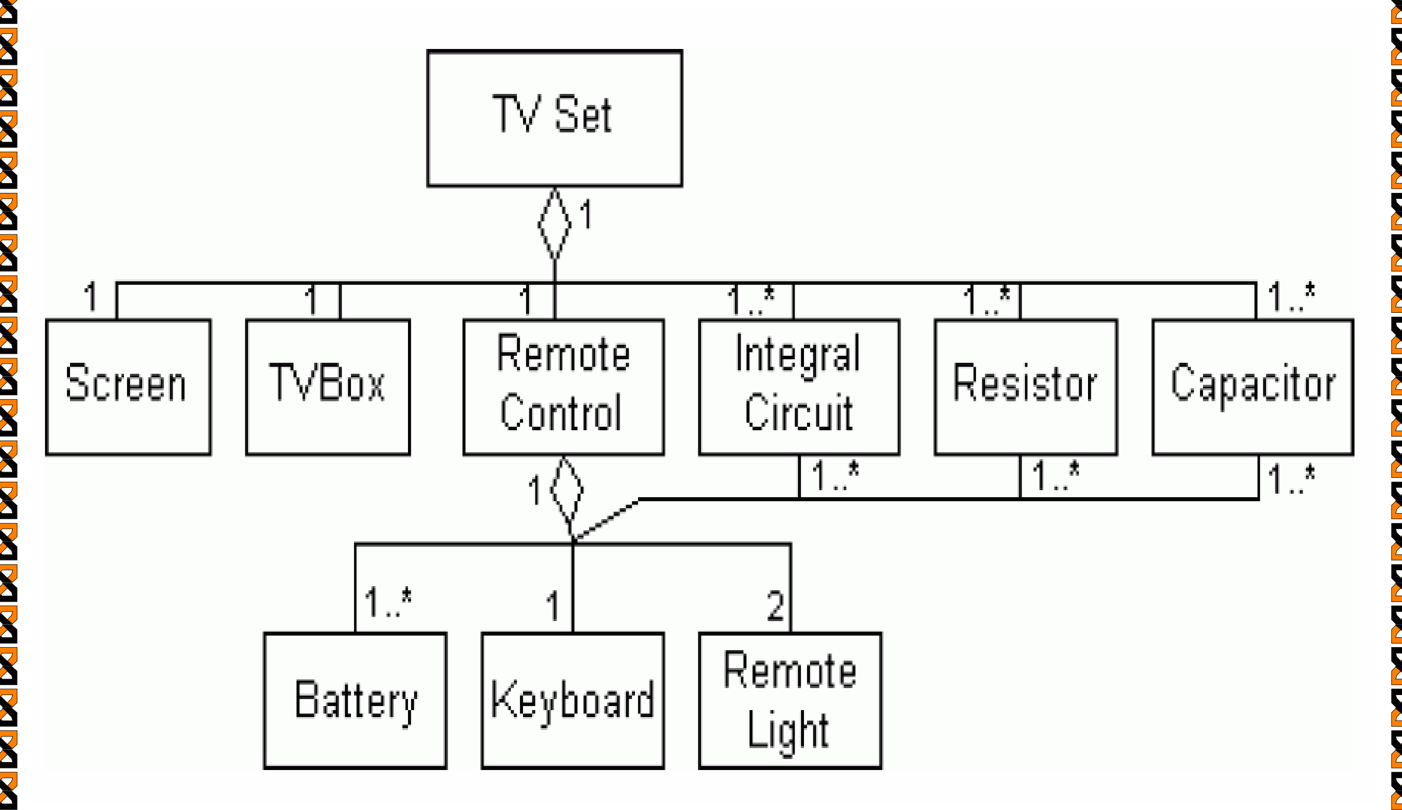
هي علاقة بين الكل والجزء ويتم تمثيلها كما يلي:

Vehicle ◇—— 1..* **Vehicle parts**

Country ◇—— 1..* **City**

تكون دورة حياة صف الجزء مستقلة عن دورة حياة صف الكل (يمكن رؤية الجزء بعيداً عن الكل).

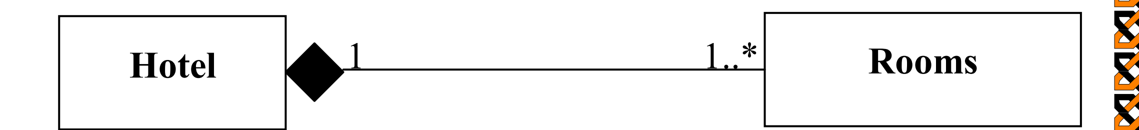
أمثلة:



علاقة Composition

هي علاقة بين الكل والجزء أيضاً لكنها أشد قسوة حيث أن دورة حياة صف الجزء تعتمد على دورة حياة صف الكل (لا يمكن رؤية الجزء بعيداً عن الكل).

يتم تمثيل هذه العلاقة كما يلي:

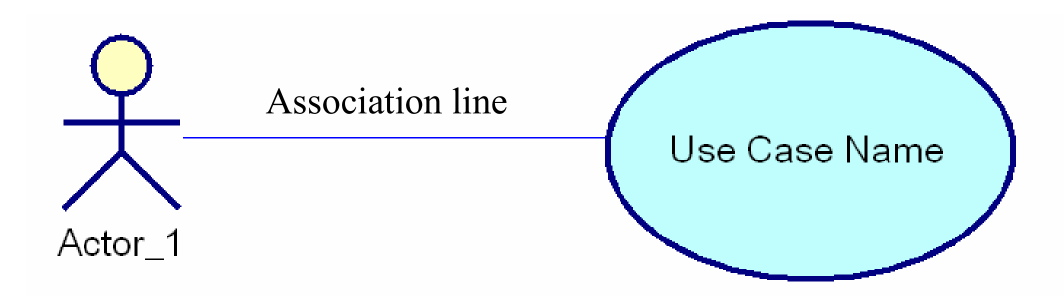


Use Case diagram

كل Use Case تمثل وظيفة من وظائف النظام وبالتالي يتضمن هذا المخطط :

- الوظائف المطلوبة من النظام.
- لمستخدم actor الذي يقوم بطلب هذه الوظيفة (قد يكون نظام آخر).

يساعدنا على بناء Use case diagram فهم سـيناريو العمـل والـذي نحـصل عليـه مـن مـستخدم النظام.



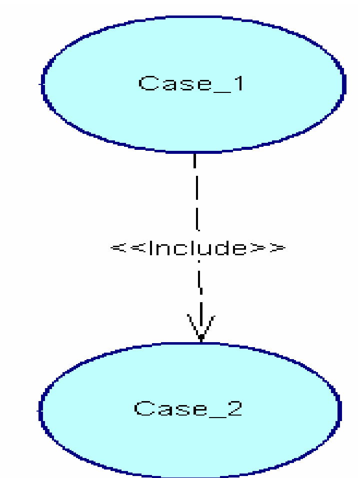
كل Use Case عبارة عن سيناريو يتكون من مجموعة من الخطوات لتنفيذ وظيفة محددة.

العلاقات في Use Case

<u>Include</u>

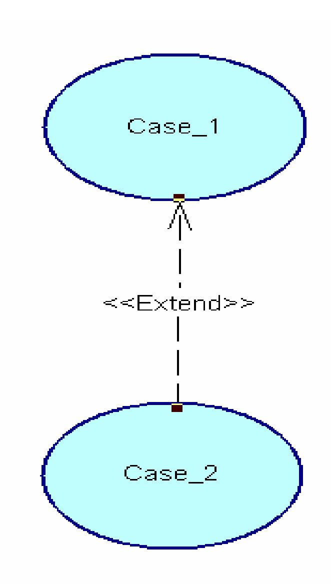
تمكننا هذه العلاقة من إعادة استخدام الخطوات الموجودة داخل Use Case

يتم تمثيل هذه العلاقة كما يلي:

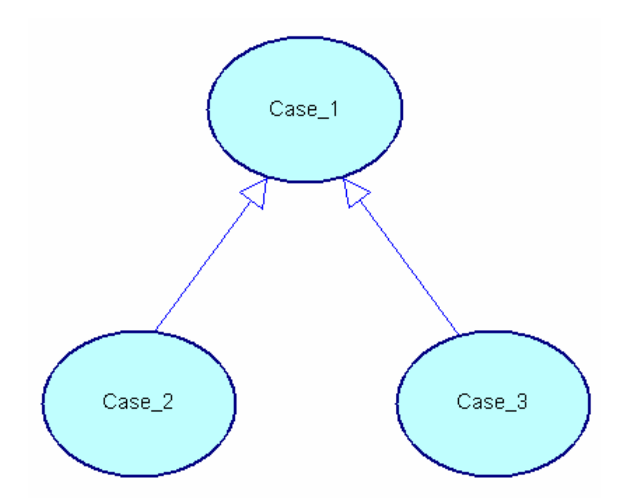


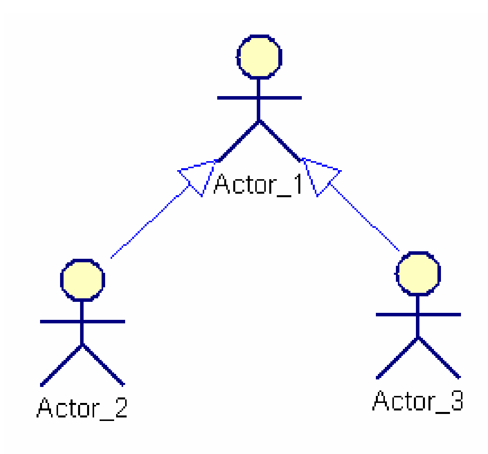
Extension

تسمح لنا هذه العلاقة بإنشاء Use Case جديدة يتم إضافتها إلى Use Case موجودة سابقا بهدف:
معالجة حالة استثنائية قد تواجهنا أو لوضع شرط على الخطوات ضمن Use Case الحالية
يتم تمثيلها كما يلي:

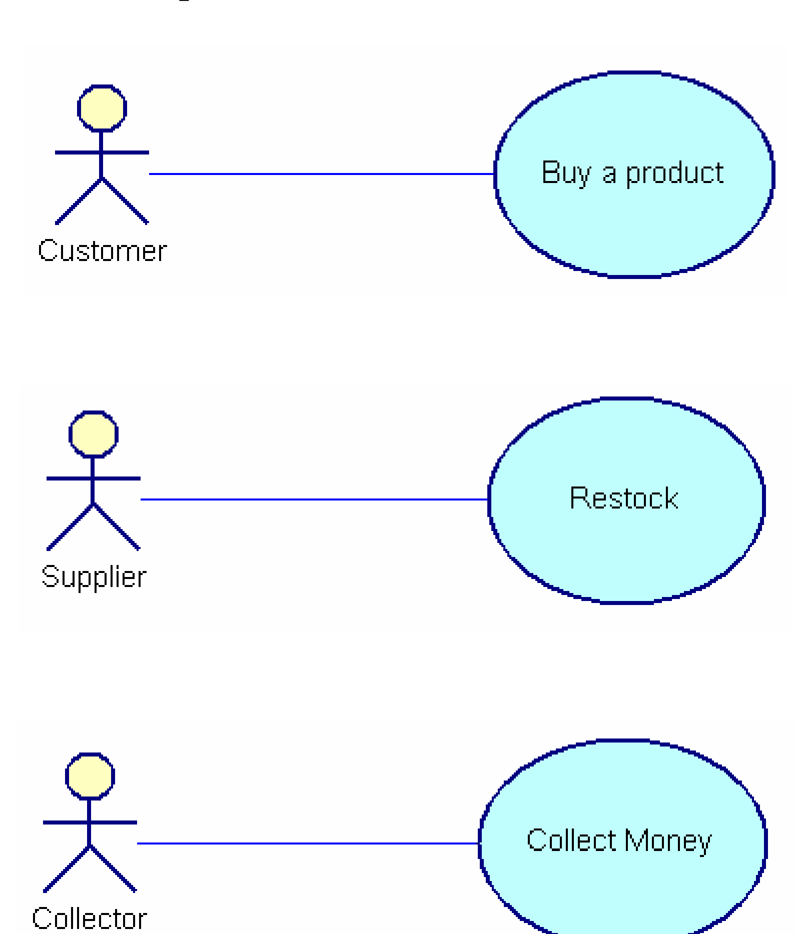


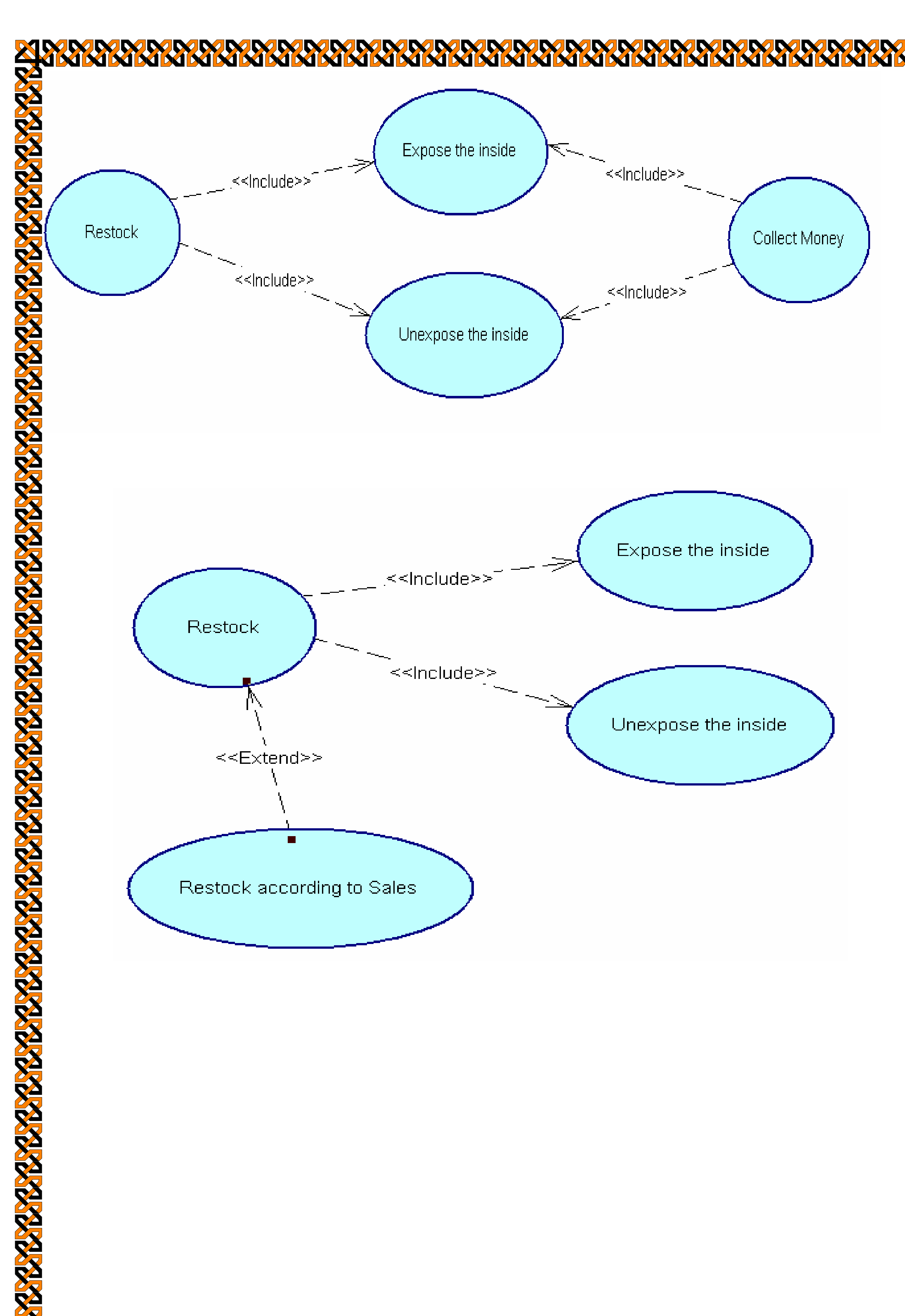
Generalization الوراثة

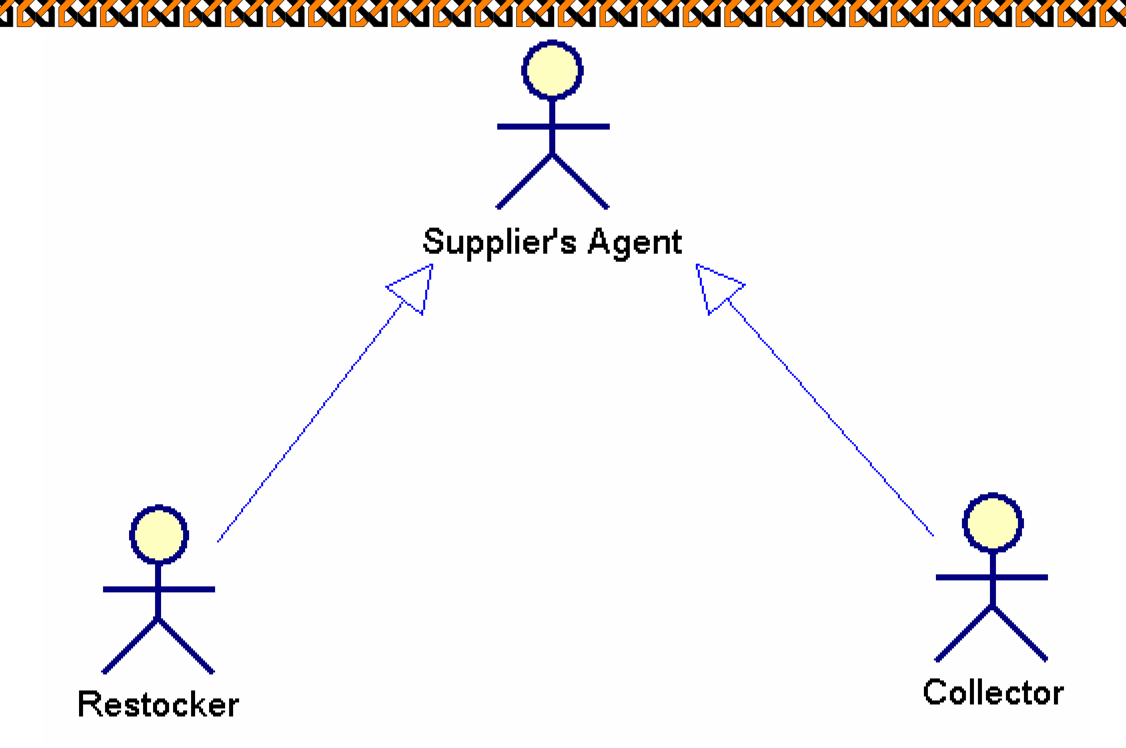




Self service Machine example



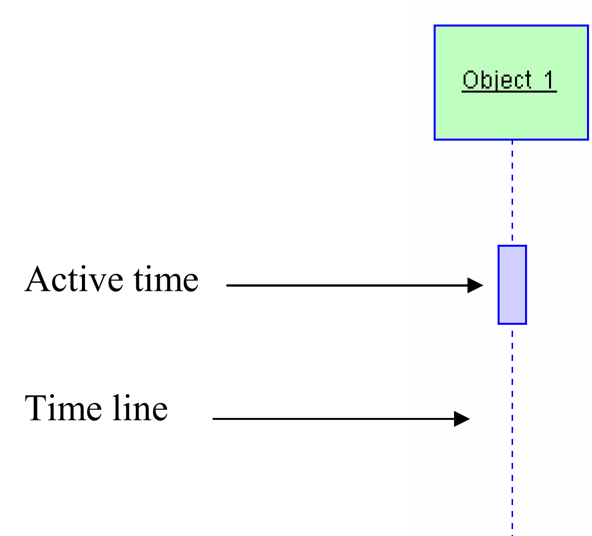

Expose the inside
<<Include>>
<<Include>>
Restock
Collect Money
<<Include>>
<<Include>>
Unexpose the inside
Expose the inside
<<Include>>
Restock
<<Include>>
Unexpose the inside
<<Extend>>
Restock according to Sales



Sequence diagram

مخطط يهدف إلى توصيف الاتصالات بين Objects عبر الزمن أي يتم إدخال بُعد الزمن Time إلى المخطط وبالتالي يتم توضيح كل التفاعلات والاتصالات بين Objects وفق تسلسل زمني.

تستخدم الرموز التالية في Sequence diagram:



توجد عدة أنواع للرسائل المتبادلة بين Objects وهي:

Message

Self message

Call message (synchronous)

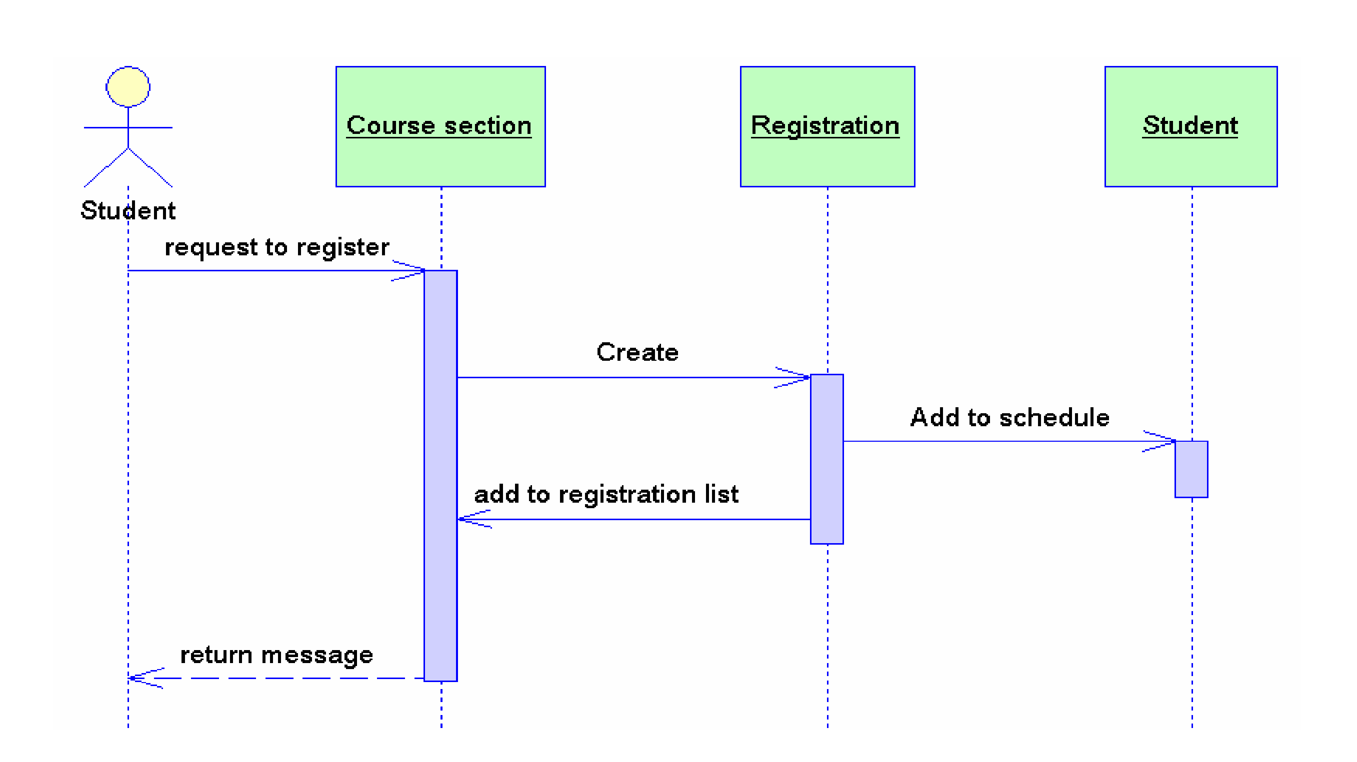
Asynchronous message

Self call message

Return message

Self return message

يوضح الشكل التالي طريقة تمثيل الرسائل في المخطط:



Actor_1

Object_1

Object_2

Message_1

Self message

<<Synchronous>>
Call message

Asynchronous message

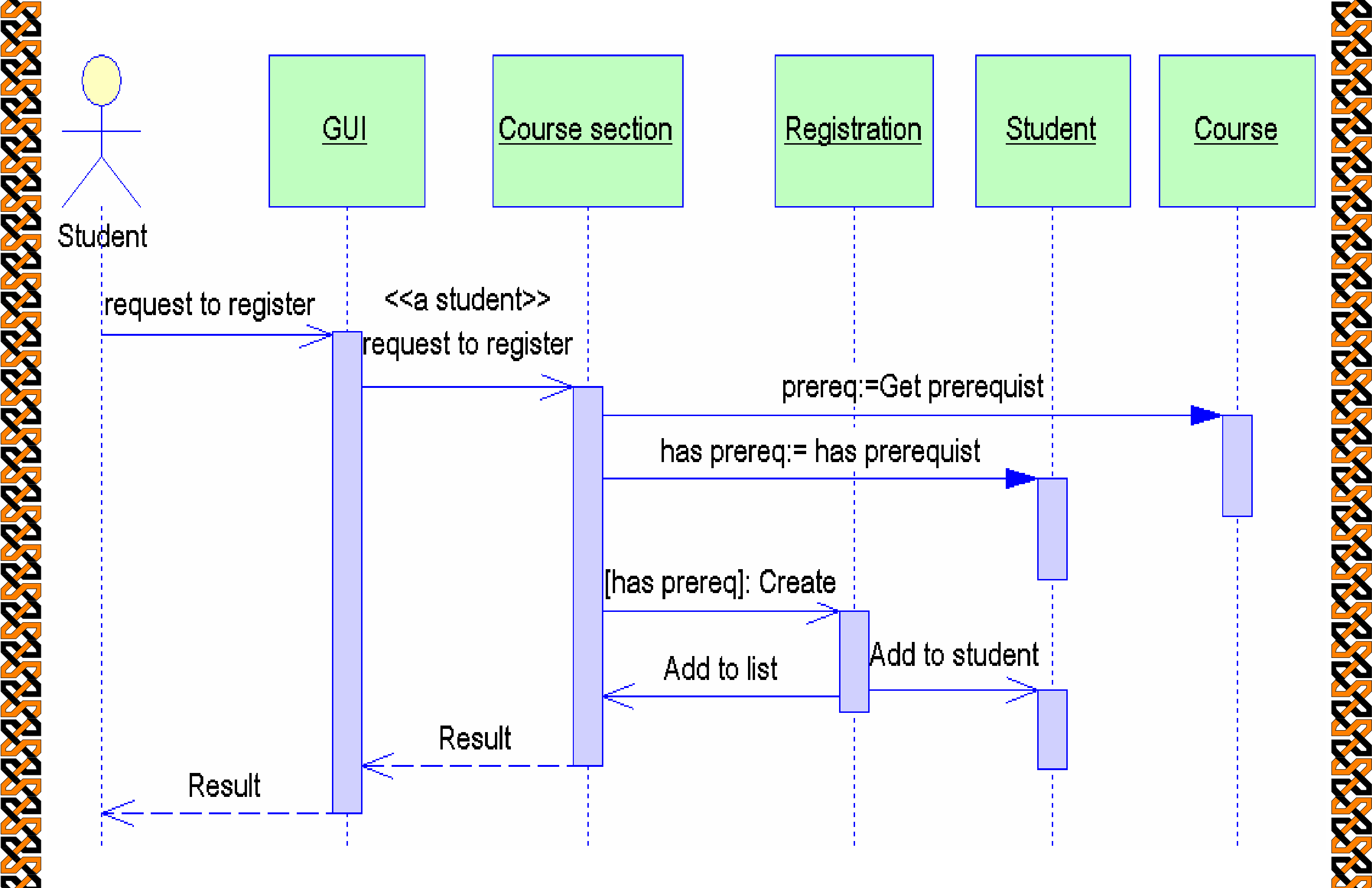
Self Call message

Return message

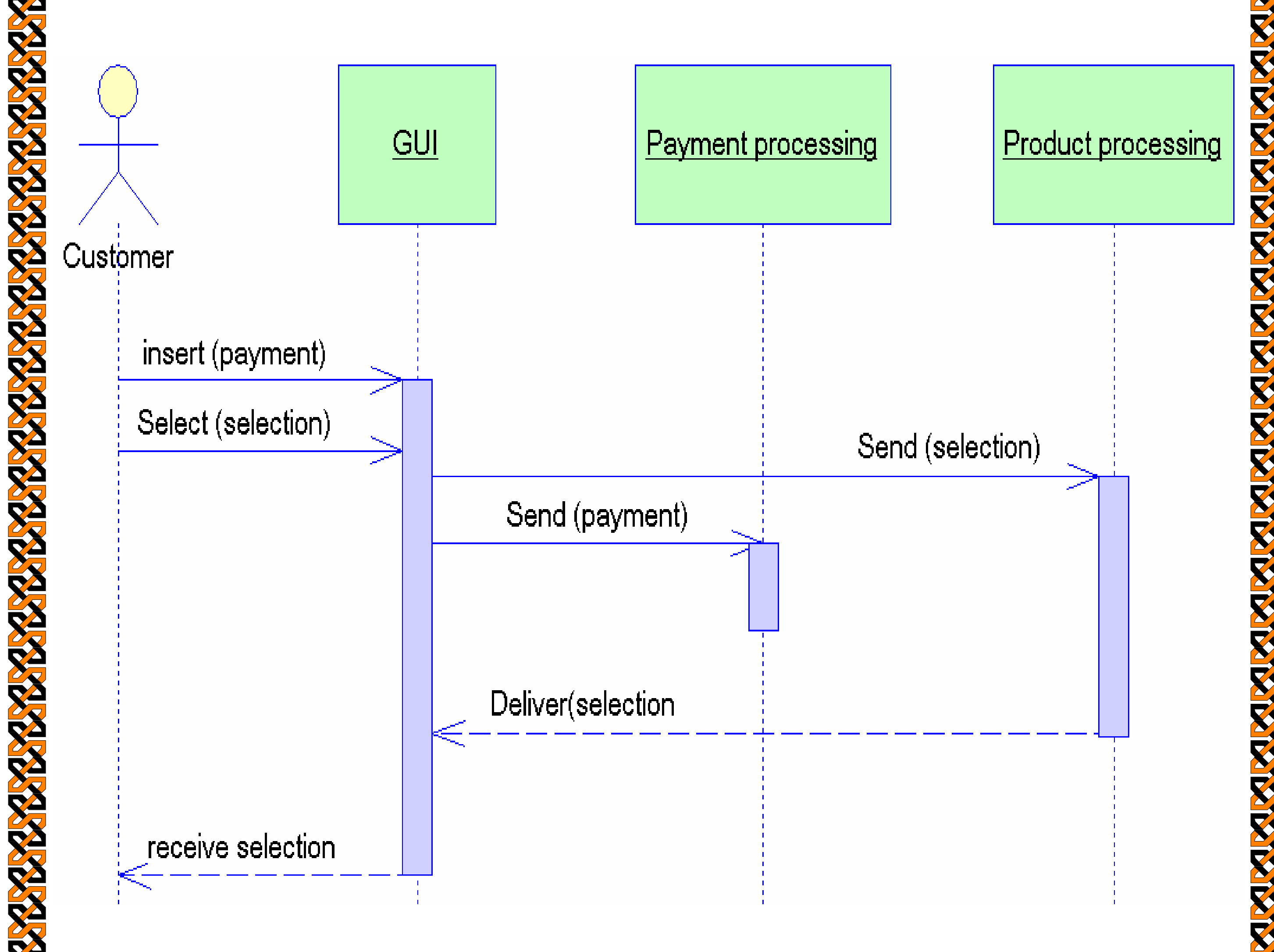
Self Return message

Return message

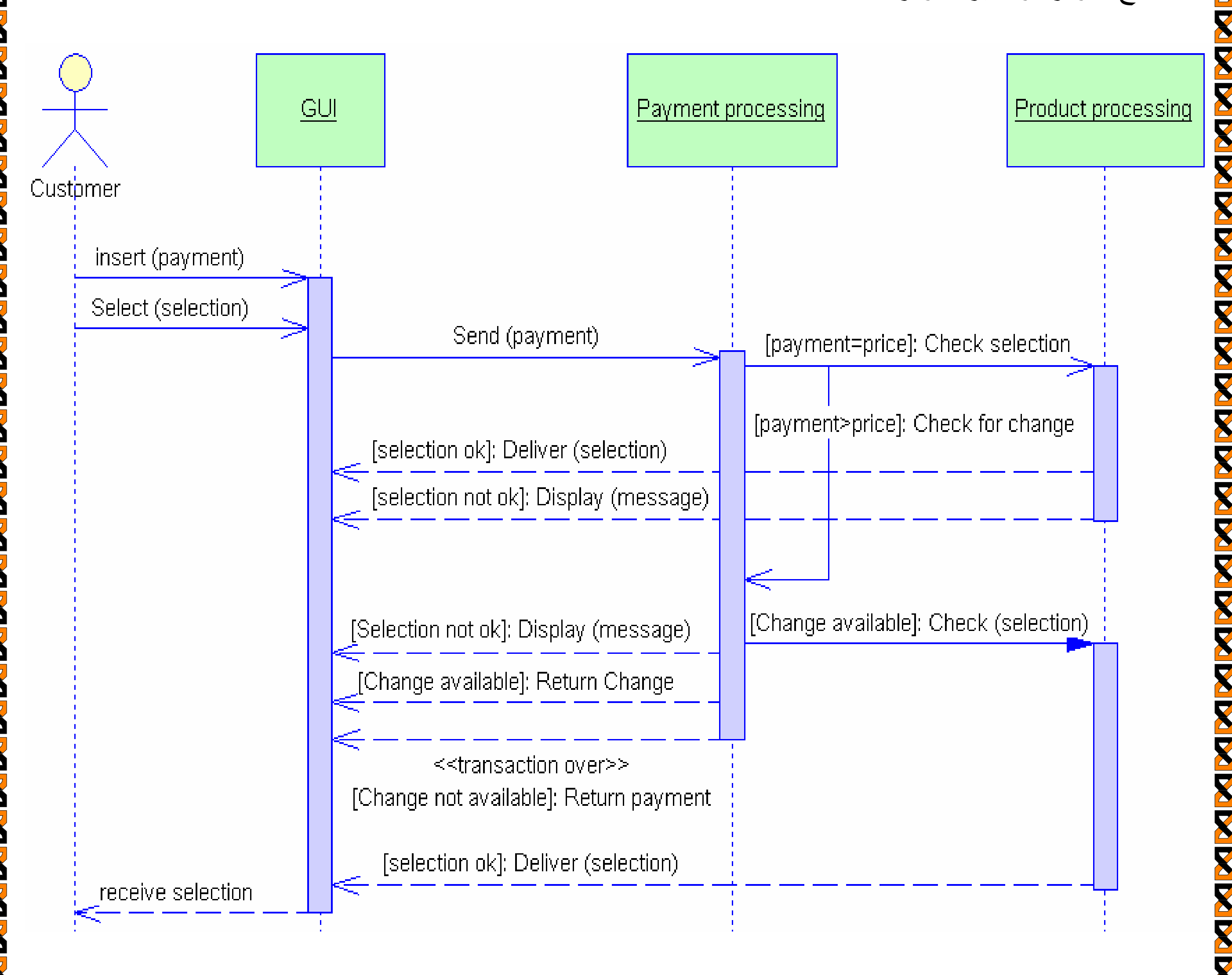
إذا أردنا تحسين النظام بحيث يأخذ بعين الاعتبار المتطلبات المسبقة لكل مادة يكون لدينا المخطط التالي:



مثال٢ : في نظام self service machine يقوم الزبون بإدخال النقود ويختار منتج فتقوم الآلة بالاستجابة لطلبه وبيعه المنتج المطلوب.



يمكن تطوير النظام بجعله يأخذ بعين الاعتبار قيمة الدفعة (اكبر أو تساوي قيمة المنتج) وأيضا فيما إذا كان المنتج متوفر أو غير متوفر.



تم بحمد الله